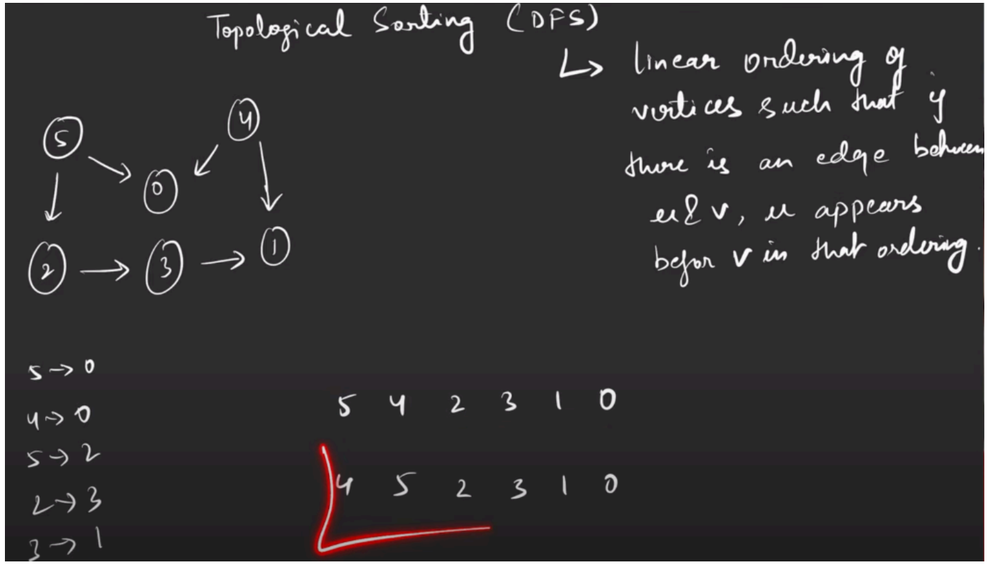
We need Directed Acyclic Graph (DAG) for topological sort




```java
class Solution
{
    //Function to return list containing vertices in Topological order.
    static int[] topoSort(int V, ArrayList<ArrayList<Integer>> adj)
    {
        // add your code here
        Stack<Integer> stk = new Stack<>();
        boolean[] vis = new boolean[V];
        for (int node = 0; node < V; node++) {
            if (!vis[node])
                dfs(adj, node, stk, vis);
        }

        int[] res = new int[V];
        int idx = 0;
        while (!stk.isEmpty()) {
            res[idx++] = stk.pop();
        }
        return res;
    }

    private static void dfs(ArrayList<ArrayList<Integer>> adj, int node, Stack<Integer> stk, boolean[] vis) {
        vis[node] = true;
        for (int ng: adj.get(node)) {
            if (vis[ng])
                continue;
            dfs(adj, ng, stk, vis);
        }
        stk.push(node);
    }
}
```

Using DFS

Another approach is using the inorder

1. Create adjacency list
2. Use BFS and add all nodes with indegree zero to initial queue
3. for each popped node for all it's neighbors decrement the indegree of the neighbor and if it's zero add it to queue
4. At the end we have traversed all the nodes in topological sorting order.